Synchro-Phasor Data Conditioning and Validation Project

Phase 2, Task 1

Report on:

Develop Error Simulation Utility

Prepared for the
Office of Electricity Delivery and Energy Reliability,
Transmission Reliability Program of the U.S. Department of Energy
Under Contract No.DE-AC03-76SF00098

Prepared by:

Electric Power Group

Ken Martin
Jianzhong Mo

Project Manager: Dejan J. Sobajic

April 26, 2014

Acknowledgement

The work described in this report was coordinated by the Consortium for Electric Reliability Technology Solutions, and funded by the Office of Electricity Delivery and Energy Reliability, Transmission Reliability Program of the U.S. Department of Energy through a contract with Electric Power Group, LLC administered by the Lawrence Berkeley National Laboratory.

Disclaimer

Preface

Synchrophasor systems have been deployed across the world. Data quality becomes a significant issue when users do not trust that it accurately represents the measured quantities. This not only causes frustration with using existing systems, but sometimes hinders further adoption of synchrophasor technology. To address the data quality problem, Department of Energy (DoE) has funded the Phasor Data Validation and Conditioning project to develop and demonstrate algorithms to detect common data quality issues in real-time (DE-AC02-05CH11231).

As part of Phase 2 task 1, this Report documents the development of Error Simulation Utility or PMU Simulator. This utility will be used to simulate phasor data and data quality errors. This utility tool will be used for Phasor Data Validation and Conditioning prototype demonstration (Phase 2).

Table of Contents

Table of Figures

Synchro-Phasor Data Validation and Conditioning Project

Phase 2, Task 1

Report on: Develop Error Simulation Utility

1. Introduction

The Synchrophasor Data Conditioning and Validation Project sponsored by the US department of Energy Consortium for Electric Reliability Technology Solutions (CERTS) program was started in December 2012. The project objectives are to develop, prototype, and test various methods for conditioning and validating real-time synchrophasor data. The project is divided into three phases.

- Phase 1: Conceptual Design and Prototype Development
- Phase 2: Prototype Demonstration
- Phase 3: Functional Specifications of the Data Validation System

In Phase 1 Electric Power Group, LLC (EPG) has completed the design and prototype development to meet the data validation and conditioning requirements. These requirements have been developed by EPG based on surveys, literature research, and experience in working with customers.

Phase 2 Prototype Demonstration has four tasks:

- Task 1: Develop Error Simulation Utility
- Task 2: Data Validation Prototype Demonstration
- Task 3: Summary Report
- Task 4: Review Meeting with Project Participants.

This report focuses on Task 1 – Develop Error Simulation Utility. The goal is to develop a utility capable of streaming PMU-type data records. This utility shall be designed to corrupt subsets of PMU data to simulate various data quality issues.

EPG has developed a utility named PMU Simulator to stream PMU-type data records. Demonstration of the algorithm and prototype developed in Phase 1 requires the ability to generate realistic data with the selected impairments. The PMU Simulator is designed to include functions for error injection and simulation.

2. Data Errors for Simulation Requirements

In Phase 1, EPG has identified seven groups (PDVC -1 to PDVC - 7) of Prototype requirements as listed in the following table. Among them, Message Error (PDVC - 1), Data Error (PDVC - 2), Topology Error (PDVC

- 4), and Communication Error (PDVC - 6) are errors that will be detected by PDVC Prototype. The error simulation utility should be able to simulate the above four types of errors.

Table 1 PDVC Functional Requirements

ID of Requirement Group	Requirements by Error Detection Type
PDVC - 1	Detect Message Errors: • Data corruption • Data tampering
PDVC - 2	Detect Data Errors • Loss of data from one or several PMUs • Loss of signals in a PMU • Offset in signal magnitude and phase • Corrupted and drifting signals in a PMU • Corrupted and drift timing time reference in one or several PMUs • Intermittent communications, inconsistent data rates and latencies • Frozen or repeated (stale) measurements • Measurement identification • Measurement corruption • Measurement exceeds engineering limits
PDVC - 3	Flag Errors & Data Correction • Data quality flags (good, suspect, bad) • Indication for other programs • Indicate safe uses for impaired data • Offer users the choice to auto-correct or only flag errors
PDVC - 4	Error Detection based on System Topology
PDVC - 5	Error Analysis & Logging • Guidance for likely error cause (and thus for resolution)
PDVC - 6	Error Detection of Communication, Network, and Program Interfaces
PDVC - 7	System Level Requirements • Standalone Application • Data Validation and Conditioning as Library

3. Data Errors Simulation Approach

PMU Simulator is able to replay recorded data, inject errors, and output error injected data as C37.118.2 which will feed PDVC prototype application. PMU Simulator is also able to simulate data errors in real-time without recorded data files. The workflow of PMU Simulator is shown in Figure 1.

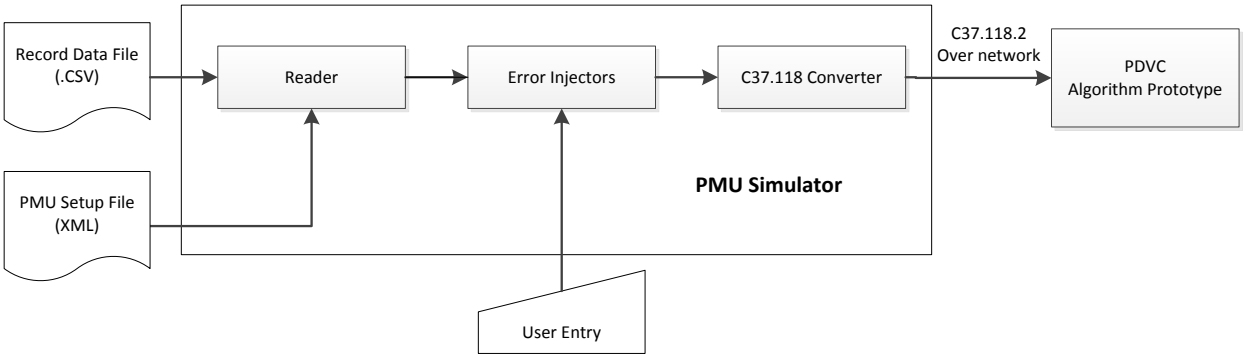

Figure 1 PMU Simulator Work Flow with PDVC Prototype

4. PMU Simulator Design

4.1 Overview

To make manual error injection easy, PMU Simulator utility is designed to be a standalone application with a Graphical User Interface. The PMU Simulator layout is shown in **Error! Reference source not found.**.

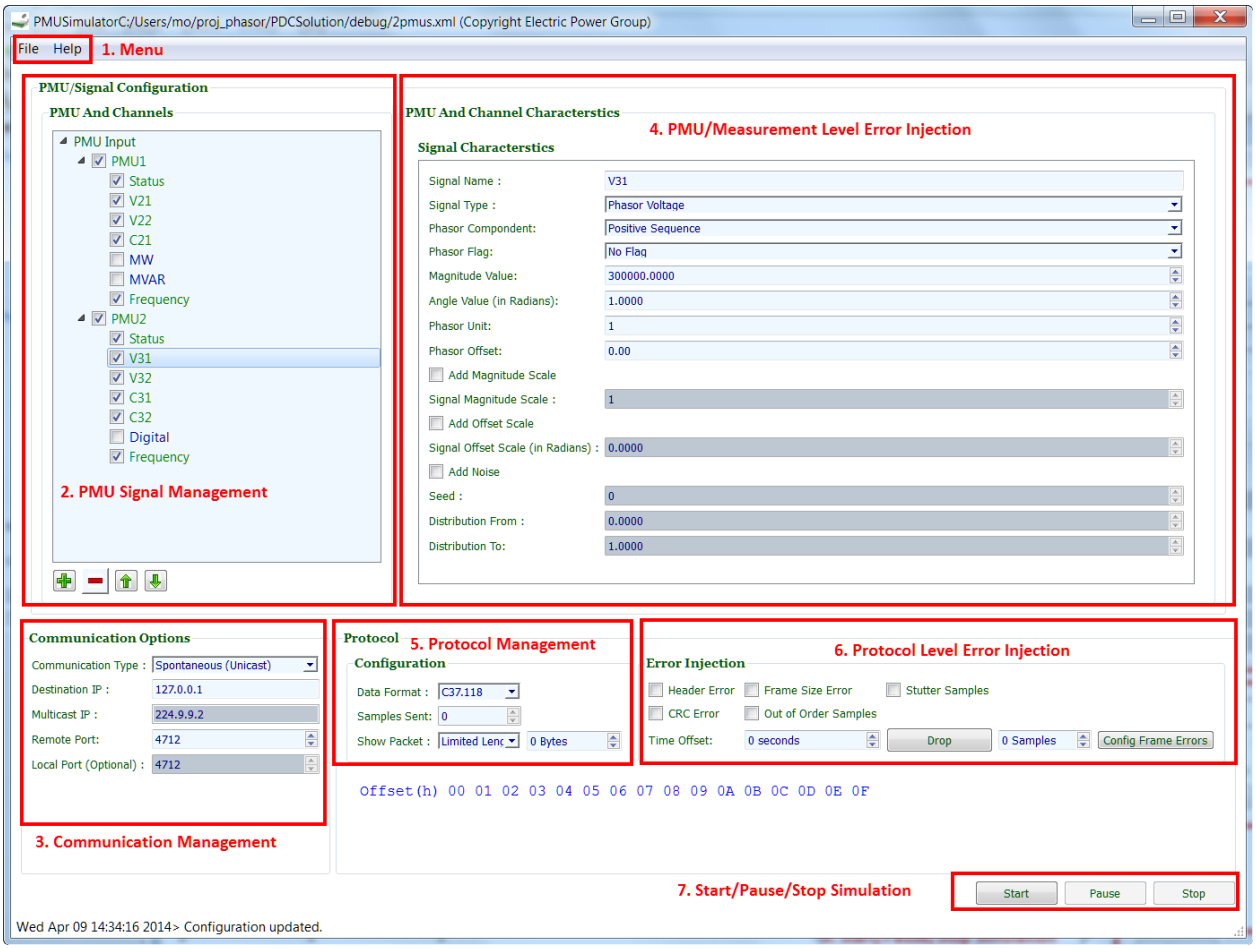

Figure 2 PMU Simulator Overview

PMU Simulator layout is divided into following areas:

- Menu – open recorded data file, open/save PMU setup file
- PMU/Signal Management – add/remove PMU/Signal for simulation
- Communication Management – TCP or UDP
- Protocol Management – C37.118 or C37.118.2
- PMU/Measurement Level Error Injection – add error to measures, change PMU Status word.
- Protocol Level Error Injection – applicable for all PMUs such as CRC error
- Start/Pause/Stop simulation

The design of PMU Simulator is to meet the error simulation requirements described in section 2. Data Errors for Simulation Requirements. Table 2 shows the error simulation functions mapping to PDVC Prototype functions.

Table 2 Error Simulation Functions and PDVC Functional Requirements Mapping

Error Simulation Functions	Requirements by Error Detection Type
Protocol Level Error Injection	Detect Message Errors: • Data corruption – corrupted through network, detectable through CRC check • Data tampering – data manipulated by unauthorized access while data is transferred on the wire, detectable through CRC, C37.118 header validation, and frame size validation with configuration frame. • Loss data for all PMUs • Intermittent communications, inconsistent data rates and latencies • Corrupted and drift timing time reference in one or several PMUs
PMU/Measurement Level Error Injection	Detect Data Errors • Loss of data from one or several PMUs • Loss of signals in a PMU • Offset in signal magnitude and phase • Corrupted and drifting signals in a PMU • Frozen or repeated (stale) measurements • Measurement identification • Measurement corruption – measurement corrupted while data is transferred on the wire, detectable through CRC check, value range check, and topology based check. • Measurement exceeds engineering limits • Topology-based error
Communication Management	Error Detection of Communication, Network, and Program Interfaces

4.2 Protocol Level Error Injection

Figure 3 shows the user interface to inject C37.118 and C37.118.2 message level errors.

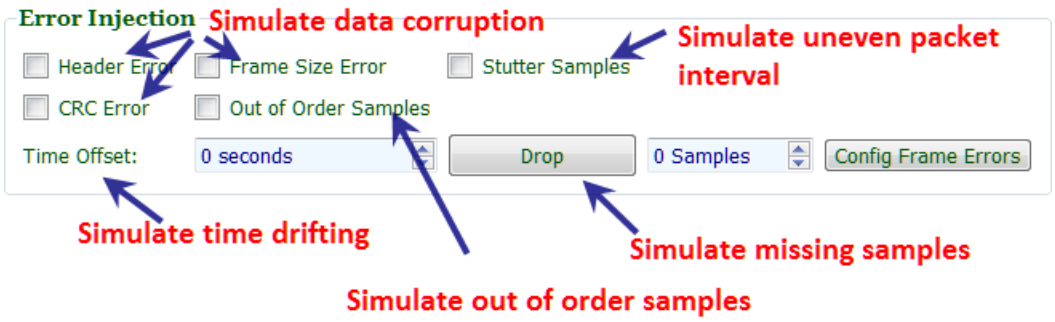

Figure 3 Protocol Level Error Injection

Three types of message level errors can be simulated:

- Data/Measurement corruption and tampering - modifying CRC, modifying frame size, and modifying C37.118/C37.118.2 sync byte for user specified number of samples.
- Loss data for all PMUs (missing data frames) – dropping user specified number of samples.
- Inconsistent data rates and latencies – enable/disable Out of Order Samples error injection.
- Intermittent communications – enable/disable Stutter Samples error injection
- Corrupted and drift timing time reference in one or several PMUs – change time offset to user specified seconds. It affects all PMUs with a data frame. A data frame can have one or several PMUs data in it.

4.3 PMU/Measurement Level Error Injection

4.3.1 PMU Level Error Injection

PMU Simulator allows user to inject PMU level errors by changing the PMU Status word as shown in Figure 4:

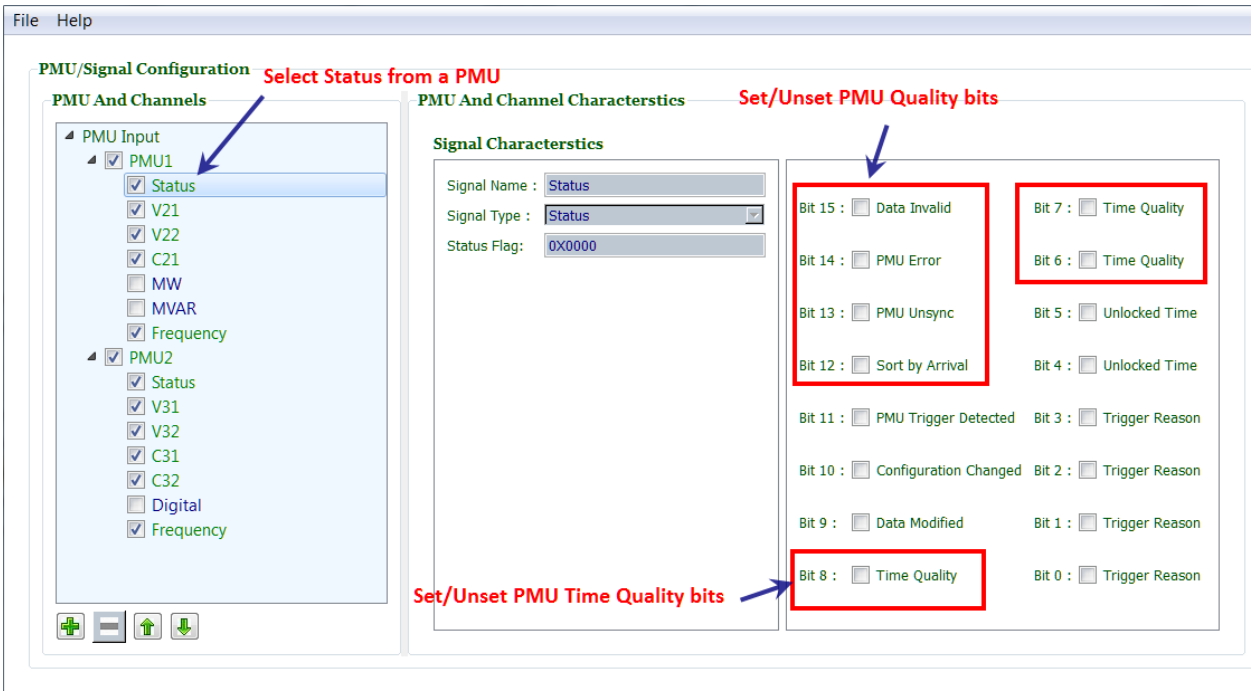

Figure 4 PMU Level Error Injection

By setting PMU Quality bits, the following errors can be simulated:

- Loss of data from one or several PMUs – set all Bit 12 – Bit 15 for selected PMU or PMUs
- Corrupted and drift timing time reference in one or several PMUs – set Bit 13 PMU Unsync for selected PMU or PMUs.
- Measurement corruption – set Bit 14 or bit 13 for selected PMU or PMUs

By setting PMU Time Quality bits, the following error can be simulated:

- Corrupted and drift timing time reference in one or several PMUs – set Bit 6 – Bit 8 for selected PMU or PMUs

4.3.2 Measurement Level Error Injection

PMU Simulator allows user to inject measurement level errors by changing the PMU Status word as shown in Figure 5.

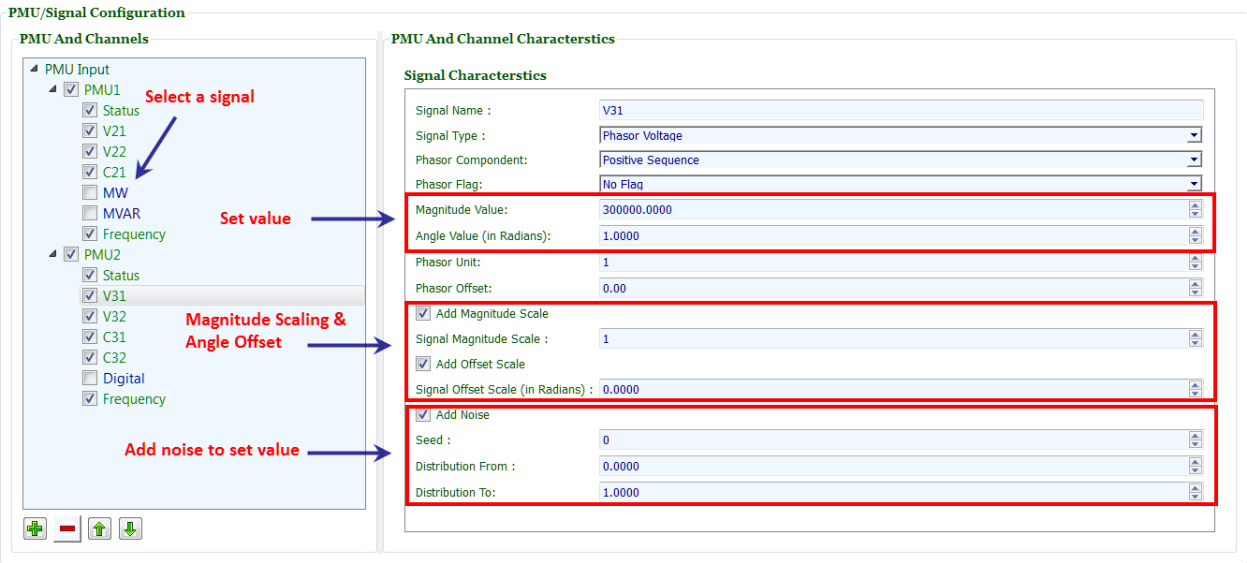

Figure 5 Measurement Level Error Injection

Measurement level error injection can simulate following errors:

- Offset in signal magnitude and phase – enable magnitude scaling and angle offset
- Frozen or repeated (stale) measurements – disable adding magnitude scaling and noise and keep the set value for long enough time with respect to the threshold used by PDVC stale detection algorithm.
- Measurement exceeds engineering limits – change the set value to high or low with respect to high/low threshold used by PDVC range check algorithm
- Measurement corruption (noisy) – enable adding noise and configure seed, distribution with respect to threshold used by PDVC noise detection algorithm. Pseudo-random number generation algorithm, a uniform random number generation algorithm, is used random number generation. "seed" is needed for algorithm initialization for the generated sequence. "distribution" is the range that random numbers will uniformly distributed within.
- Topology error – set values of all signals involved in the PDVC topology definition so that the calculated results based on topology definition violates the threshold used by PDVC topology detection algorithm.

4.4 PMU/Signal Management

PMU Simulator allows users to add/remove PMUs and add/remove signals within a PMU. It also allows users to change PMU/Signal attributes such as PMU Station Name and signal Channel Name. Figure 6 shows the user interface for PMU/Signal management.

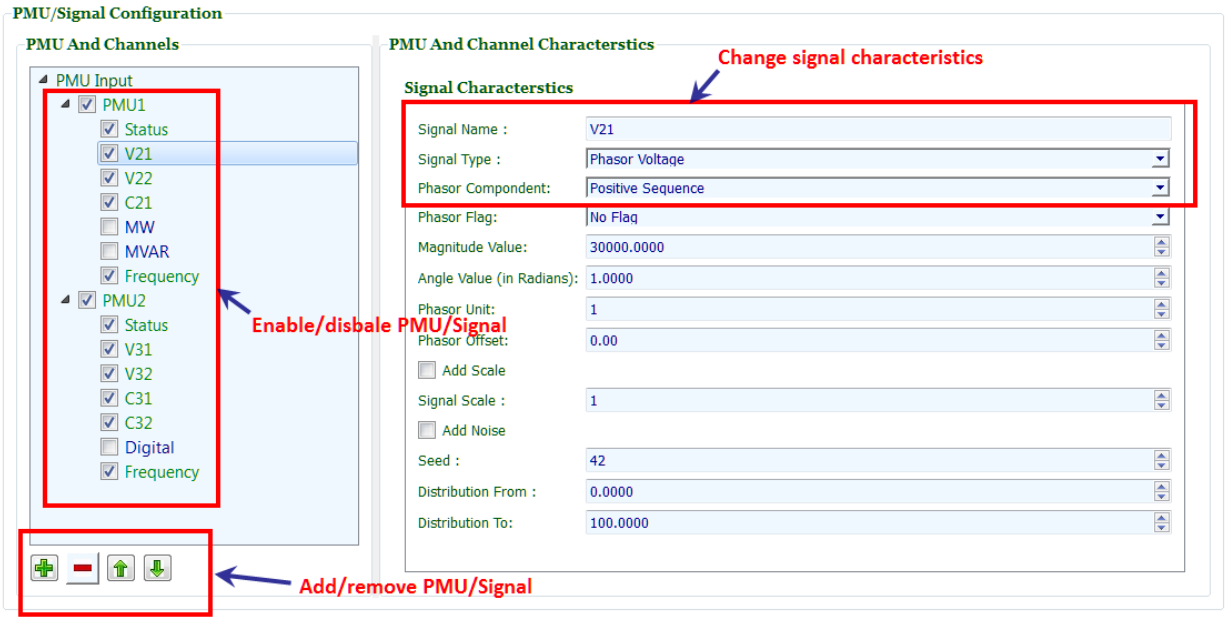

Figure 6 PMU and Signal Management

Through PMU/Signal management, following situations can be simulated:

- Loss of data from one or several PMUs – disable one or more PMUs
- Loss of signals in a PMU – disable one or more signals within a PMU
- Measurement identification – change PMU Station Name or change signal Channel Name

4.5 Recorded File Simulation

PMU Simulator can replay data records stored as EPG's Phasor Data CSV File Format (version 3). After loading the CSV file, users can apply magnitude scaling factor, angle offset, and noise to signals to inject errors as described in section 4.3.

EPG's Phasor Data CSV File Format (version 3) specification is attached along with this report.

4.6 Communication Management

PMU Simulator supports all types of TCP/IP communication options as shown in Figure 7. Setting the communication option different from PDVC Prototype can simulate communication error.

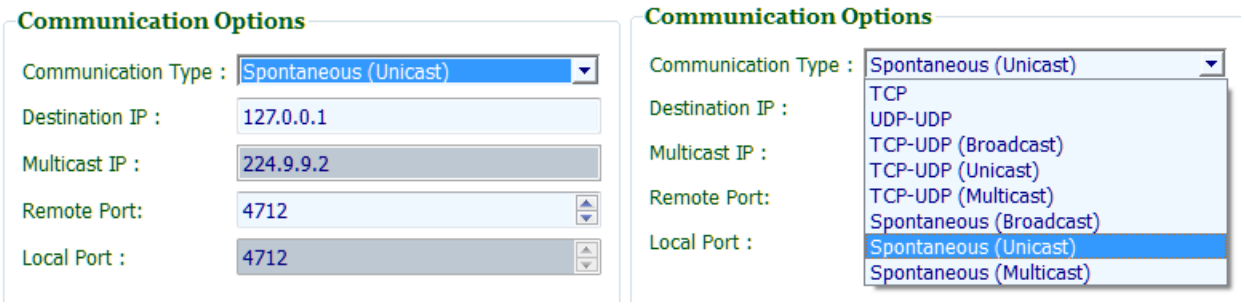

Figure 7 Communication Management

5. Conclusion

EPG has developed the utility named PMU Simulator to simulate errors which PDVC algorithms are designed to detect. PMU Simulator has been used for PDVC Prototype factory testing. PMU Simulator will be used for PDVC demonstration using recorded data – provided by BPA and PJM as required by Phase 2 Task 2: Data Validation Prototype Demonstration.

Appendix: EPG Synchrophasor Data File Format (.CSV)

Version 3, 2012

1. Introduction

Synchrophasor data is used both in real-time from direct transmission and after the fact as recorded data. The SYNCHROPHASOR standard, C37.118, describes a real-time data transmission format but does not define a format for recorded data storage or exchange. Several formats have been used for synchrophasor data, most notably the Phasor File (also known as 'dst') data format pioneered in the WECC and supported by Bonneville Power Administration users.

The IEEE COMTRADE standard is a file format designed for time series data that is established worldwide and is supported by standards making bodies. It has a significant number of recording parameters that have been adapted for phasor data. COMTRADE file format is getting adopted quickly.

Phasor File and COMTRADE formats are binary files. It is not human friendly and hard to manipulate their content. It's desirable to have a human readable and easy to manipulate file format for data exchange. The CSV (comma separated values) file format is a widely accepted format to store data in ASCII files in which data is stored in rows and columns. The data in rows are and delimited by comma (,). These files are human readable, and can be easily recognized by spreadsheet applications and other applications to perform data analysis and manipulation. This document proposes a format to store Synchrophasor data in CSV files, which is adopted by Electric Power Group's synchrophasor applications, including PGDA, RTDMS, ePDC, and PMU Simulator. All EPG applications shall support importing/exporting synchrophasor data in this format.

Section 2 describes the CSV file naming convention. It is highly recommended for event data, though not required in many cases. Section 3 describes the header and data arrangement in the CSV file.

2. CSV File Naming Convention

The file name follows the convention defined in **IEEE PC37.232** standard. This file name convention defines a readable, comma delimited filename format. The "required fields" for the filename shall be as follows and in order as shown here:

Start Date, Start Time, Time Code, Station ID, Device ID, Company Name,

Additional fields may be added as needed and are called "user fields" are shown here:

User 1, User 2, User3, and so on .Extension

2.1 Start Date Field Format

This field is a fixed numeric format of (yymmdd). The start date is the date at which the first sample was recorded or the date at which the first trigger occurred. This is determined in the Time Code field.

2.2 Start Time Field Format

This field is a non-fixed numeric format and can be specified to the required precision (hhMMssmmmuuu). The start time is the time at which the first sample was recorded or the time at which the first trigger occurred, also determined in the Time Code field.

2.3Time Code Field Format

This field is limited to a maximum of seven characters. The first character is the sign and is followed by up to five characters indicating the time difference between the time system used for the Start Date and Start Time fields (collectively, the Time Tag) and Universal Coordinated Time (UTC) without offset. The format for the five characters is as follows and in order: up to two digits for the hours followed by the letter "h" followed by two digits for the minutes. The last three characters are required only when fractional hours are in use. In addition, one final character, the letter "t", should be concatenated at the end to the Time Code Field if the filename's Time Tag is referencing the date and time of the first trigger. The calculation for the time difference should also consider whether standard time or daylight time was in affect at the time of the recording. The following examples provide a number of valid descriptions for the Time Code Field:

-4 *Time Tag is 4 hours behind UTC*

+5t *Time Tag is 5 hours ahead of UTC and references trigger time*

-7h15 *Time Tag is 7 hours and 15 minutes behind UTC*

0t *Time Tag is UTC and references trigger time*

2.4 Station Identifier, Device Identifier, and Company Name Fields Formats

Users may formulate their own codes for these fields.

The following fields are optionally recommended for packing meaningful information in the filename:

Duration Field Format:

The duration is equal to the time difference between the first and last samples in the file. The fields should be in the same format as the Start Date and Start Time Fields. For example a file with 2 years, 1 day, and 99 seconds of data will have a duration code of "020001,000139".

Type Field Format:

This field may be used to describe the type of originating event and users may formulate their own codes. For example, the code "AG" may be used for a record created from a phase A to ground fault event.

Geographic Position Coordinates Field Format:

Two user fields are required in order to support position information. The first field is an expression of latitude and the second field is an expression of longitude.

2.5 Examples of File Names

- `120831,175215183,-4t,sta80,ben717,epg.csv`
  Note: The Start Time field is given in millisecond resolution and refers to when the first trigger in the record occurred with a COMTRADE configuration file extension.

- `120831,175215183,-4t,sta80,ben717,epg,000000,001359,uf,critical-frequency.csv`
  Note: This is a same example as above but with two user fields added and file duration fields added. "uf" added to represent that the file was created by a under frequency trigger and "critical-frequency" trigger also occurred in the record.

3. CSV file column header information

The first row contains column header information. The data block starts from the second row and continues to end of file.

3.1 The column header information format

The first row is the header and must start with Date Time and contain at least one signal column and ended with cartridge return and line feed character:

DateTime(ISO Format),SignalName1,SignalName2,SignalName3,… *<CR/LF>*

3.2 Signal Name format

A signal name is comprised of PMU Name, Channel Name, and signal type which are separated by dots (.).

<PMU Name>.<Signal Channel Name>.<Signal Type>

Examples:

```
Bull Run.Frequency.FR

Bull Run.Frequency.DF

Bull Run.500 kV Bus +SV.VM

Bull Run.500 kV Bus +SV.VA

Bull Run.Watts Bar +SI.IM

Bull Run.Watts Bar +SI.IA
```

```
      Bull
Run.B16;B15;B14;B13;B12;B11;B10;B09;B08;B07;B06;B05;B04;B03;B02;B01.DG
```

Note: A Digital signal has 16 bits in C37.118 standard. There are 16 channel names for a Digital signal to present every bit. These 16 channel names will be separated by semi column where the first channel represents the highest bit (bit 15) and last channel represents the lowest bit (bit 0) . For example:

```
Bull Run.B16;B15;B14;B13;B12;B11;B10;B09;B08;B07;B06;B05;B04;B03;B02;B01.DG
```

Note: There is no name for Status flags in C37.118. `Status` should be used as the ***Signal Channel Name***. For example:

```
      Callaway.Status.ST
```

Note: There is no name for frequency in C37.118. `Frequency` should be used as the ***Signal Channel Name***. For example:

```
      Callaway.Frequency.FR
```

3.3 Signal Type format

The following code describes each signal type along with its unit:

Label	Description	Engineering Unit
FR	Frequency	Hz
DF	DF/DT	Hz/s
VM	Voltage magnitude (line to line voltage)	kV
VA	Voltage angle wrapped to (-180 180] degrees	Degree
IM	Current magnitude (line current)	A
IA	Current angle wrapped to (-180 180] degrees	Degree
PP	Active power	MW
PQ	Reactive power	MVar
DG	Digital	N/A
ST	Status flag, a 16-bit unsigned integer mapped to C37.118 STAT word.	N/A

AN	Analog	Depending on the signal
ZZ	Generic type other than specified above, such as for derived/calculated values	depending on the signal
UN	Unknown or missing information	depending on the signal

3.4 Column Header Row Example

```
Date Time,Callaway.Status.ST,Callaway.Frequency.FR,Collinsville.500 kV
Line.VA,Cordova.Line1.IM
```

4. Data information

4.1 Date Time Data in ISO 8601 Format

The first column is the Date Time information in human readable format following ISO 8601 standard. Exactly the components shown here must be present, with exactly this punctuation. Note that the "T" appears literally in the string, to indicate the beginning of the time element.

```
Year:

   YYYY (eg 1997)

Year and month:

   YYYY-MM (eg 1997-07)

Complete date:

   YYYY-MM-DD (eg 1997-07-16)

Complete date plus hours and minutes:

   YYYY-MM-DDThh:mmTZD (eg 1997-07-16T19:20+01:00)

Complete date plus hours, minutes and seconds:

   YYYY-MM-DDThh:mm:ssTZD (eg 1997-07-16T19:20:30+01:00)

Complete date plus hours, minutes, seconds and a decimal fraction of a
second

   YYYY-MM-DDThh:mm:ss.sTZD (eg 1997-07-16T19:20:30.45+01:00)
```

where:

```
YYYY = four-digit year

MM   = two-digit month (01=January, etc.)

DD   = two-digit day of month (01 through 31)

hh   = two digits of hour (00 through 23) (am/pm NOT allowed)

mm   = two digits of minute (00 through 59)

ss   = two digits of second (00 through 59)

s    = three or four digits representing a decimal fraction of a second

TZD  = time zone designator (Z or +hh:mm or -hh:mm)
```

There are two ways to handle time zone offsets:

1. Times are expressed in UTC (Coordinated Universal Time), with or without a special UTC designator ("Z").
2. Times are expressed in local time, together with a time zone offset in hours and minutes. A time zone offset of "+hh:mm" indicates that the date/time uses a local time zone which is "hh" hours and "mm" minutes ahead of UTC. A time zone offset of "-hh:mm" indicates that the date/time uses a local time zone which is "hh" hours and "mm" minutes behind UTC.

Note: to preserve high enough resolution, three or four digits are used to represent the decimal fraction of a second.

4.2 Signal Data

Signal data starts at the second column and continues till (n+1) column where n is the number of signals. For Status and Digital signals, data should be saved as 16 bits unsigned integer. For phasors, analogs, frequency, and MW/MVAR signals, values are in floating point. The signal data is required to be evenly spaced ordered by increasing date/time. If the data is unavailable for any signal for a particular time, that value should be set to `-9999`.

Example of data for frequencies:

```
2010-10-08T04:50:00.0000,59.994,59.994,59.994

2010-10-08T04:50:00.0333,59.995,59.992,60.003

2010-10-08T04:50:00.0667,59.994,59.995,59.991

2010-10-08T04:50:00.1000,59.993,59.992,59.975

2010-10-08T04:50:00.1333,59.995,59.992,60.006

2010-10-08T04:50:00.1667,59.994,59.991,59.993
```